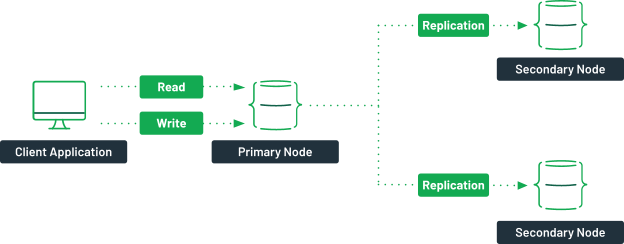
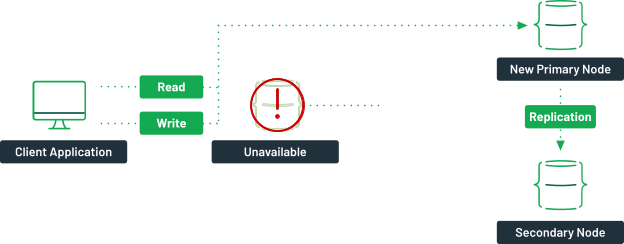
A MongoDB replica set is a group of one or more servers containing the exact copy of the data. While it’s technically possible to have one or two nodes, the recommended minimum is three. A primary node is responsible for providing your application’s read and write operations, while two secondary nodes contain a replica of the data.



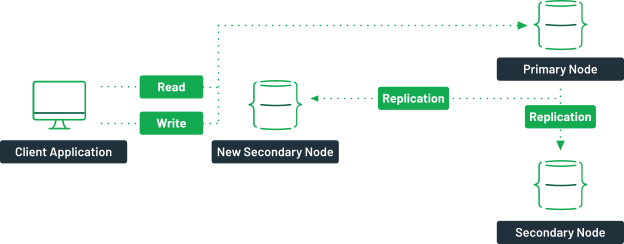
*A typical replica set in MongoDB.*

Should the primary node become unavailable for some reason, a new primary node would be picked by an [election process](https://docs.mongodb.com/manual/core/replica-set-elections/). This new primary node is now responsible for the read and write operations.



*If a primary node is unavailable, the traffic from the client application is redirected to a new primary node.*

Once the faulty server comes back online, it will sync up with the primary node and become a new secondary node in the cluster.



*When the previous primary node comes back online, it comes back as a secondary node.*

The goal is to provide your application with high availability over your data. Even in a server failure, your client application can still connect to the cluster and access the data, reducing the overall potential downtime.

Steps To Setup Mongo db Cluster

Stop the mongo process in all instances

*sudo service mongod stop*





Then Edit the file /etc/mongod.conf in all instances

# network interfaces

net:

port: 27017

bindIp: 0.0.0.0 # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, use the net.bindIpAll setting.

#security:

#operationProfiling:

replication:

replSetName: "rs1"

Start mongo service in all instances

sudo service mongod start

Then inside the mongo shell

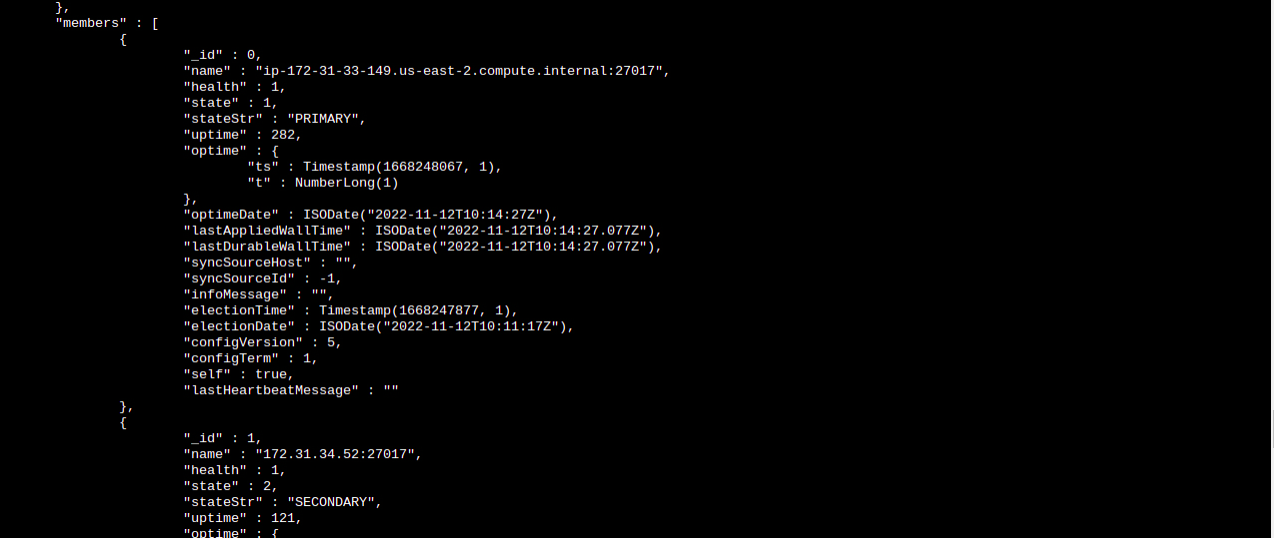
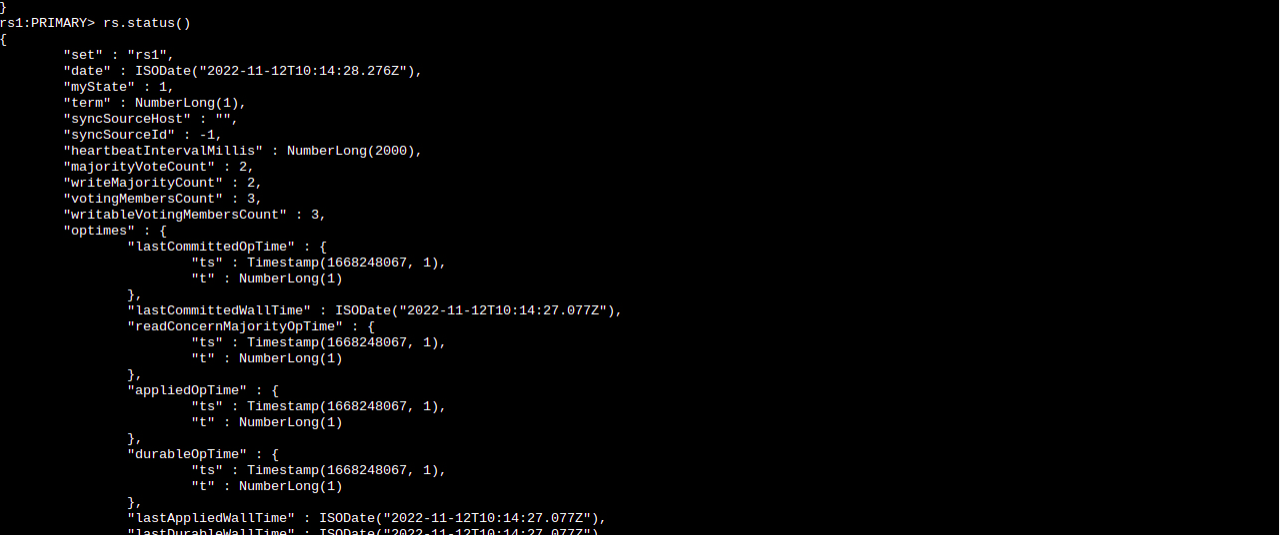
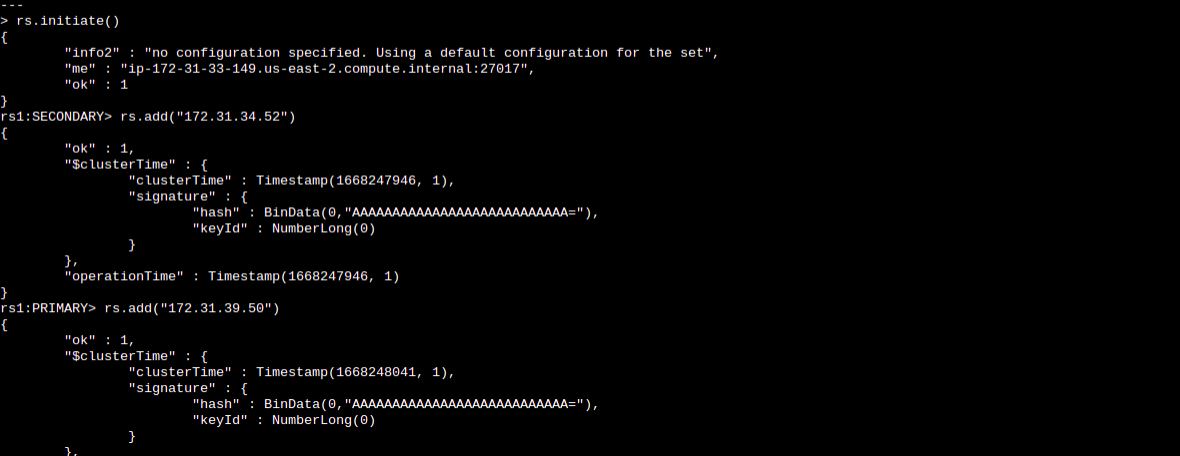
Initiate the mongodb primary node

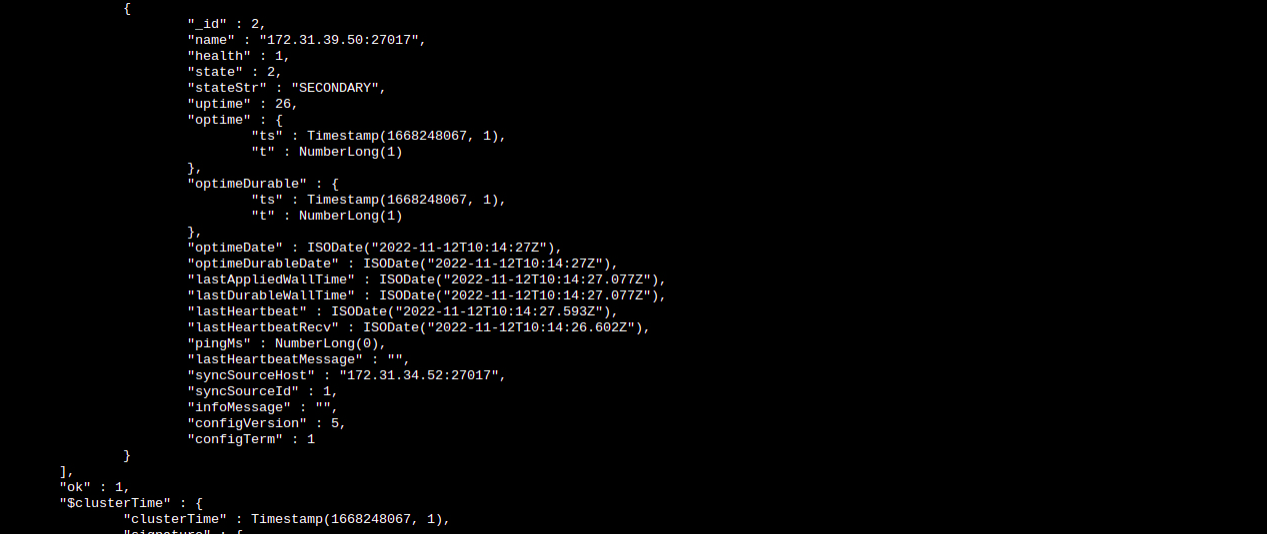
Then add Secondary nodes

> rs.initiate()

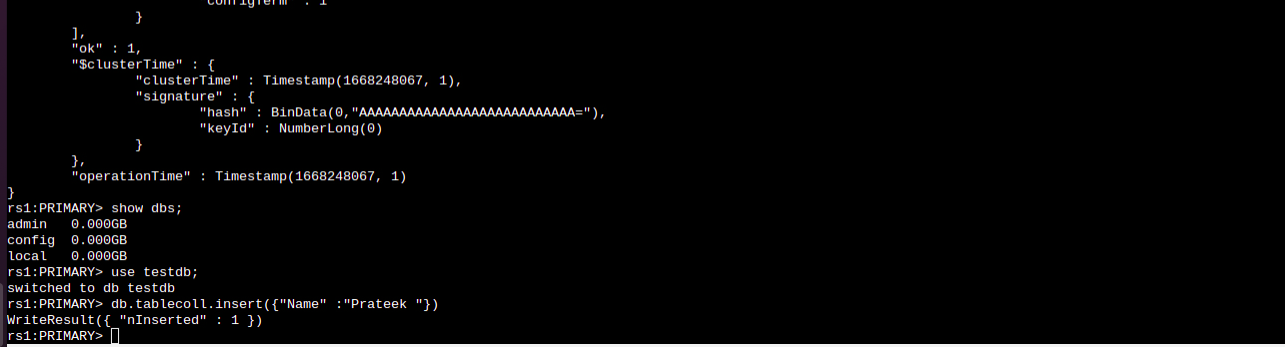
rs1:SECONDARY> rs.add("172.31.34.52")

rs1:PRIMARY> rs.add("172.31.39.50")

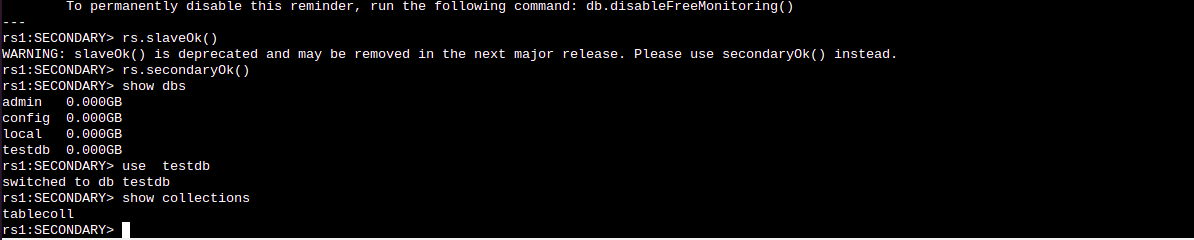




Creating database in primary node for validating



Verifying the database in secondary or slave node



Creating Cluster Through Terraform

Create the user data userdata.sh file and use the following command to edit the mongod.conf file in the primary cluster

sudo sed -i 's/127.0.0.1/0.0.0.0/' /etc/mongod.conf

sudo echo -e 'replication:\n replSetName: "rs1"' >> /etc/mongod.conf

sudo service mongod restart

mongo <<EOF

rs.initiate()

rs.add(“private\_ip of first cluster instance”)

rs.add(“private\_ip of second cluster instance”)

In the instance of Secondary cluster add the following through userdata.sh

sudo sed -i 's/127.0.0.1/0.0.0.0/' /etc/mongod.conf

sudo echo -e 'replication:\n replSetName: "rs1"' >> /etc/mongod.conf

sudo service mongod restart

Doing this will create ReplicaSet cluster of 3 nodes

And the mongod.conf inside all the clusters will look like this

**# mongod.conf**

**# for documentation of all options, see:**

**# http://docs.mongodb.org/manual/reference/configuration-options/**

**# where to write logging data.**

systemLog:

destination: file

logAppend: true

path: /var/log/mongodb/mongod.log

**# Where and how to store data.**

storage:

dbPath: /var/lib/mongo

journal:

enabled: true

**# engine:**

**# wiredTiger:**

**# how the process runs**

processManagement:

fork: true # fork and run in background

pidFilePath: /var/run/mongodb/mongod.pid # location of pidfile

timeZoneInfo: /usr/share/zoneinfo

**# network interfaces**

net:

port: 27017

bindIp: 0.0.0.0 # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, use the net.bindIpAll setting.

#security:

#operationProfiling:

#replication:

**# replSetName: "rs1"**

#sharding:

**## Enterprise-Only Options**

#auditLog:

#snmp:

replication:

replSetName: "rs1"